MARITIME BUSINESS SCHOOL
at HSBA

MARCO POLO
NEW WAYS TO A GREEN HORIZON

X-Posse Optimization Toolbox
User's Manual

ASSIST.PROF. CH.N. STEFANAKOS
PROF. O. SCHINAS

Hamburg

October 2012

This educational material is prepared for the needs of trainees of the project X-Posse, that was partially financed by the Marco Polo II Programme of the European Commission.

CONTENTS

CHAPTER 1

INTRODUCTION

X-Posse [1] is a Common Learning Action (CLA) project, partially financed by the Marco Polo II Programme of the European Commission, that aims to promote green logistics alternatives through focused training actions on sea-river, sea-rail and marketing of these services.

Although it is easy to conceive the concept of sea-river and sea-rail operations and their impact on the environment, there is no training or "how-to" material that demystifies the concept. Education and training are still very focused on the individual modes, thus interoperability is not passed into the training context. Moreover, the positive environmental impact of such solutions is understood by market forces. Hence there is a clear marketing gap, i.e. it is difficult to translate the benefits into an attractive service package.

X-Posse project will seek to fill in these gaps, being focussed on strategic actions to support the sustainable development of freight transport in cities.

Ports are becoming more and more knowledge based environments which need tailor-made and innovative education and training actions in order to support the sustainable development of intermodality and hence the seamless integration between the 'sea/port' side and the 'land' side throught 'greener' alternatives than road, such as rail and river.

X-Posse is therefore focussed on green logistic alternatives in order to ensure seemless integration of modes of transport (sea-river, sea-rail) throught new education and training concepts for interoperability. In doing so, X-Posse will address and offer solutions, ideas and business proposals through training and understanding.

The present manual presents an Optimization Toolbox specially designed and developed for the needs of X-Posse project in MATLAB$^{©}$ programming language. In Chapter 2, a detailed description of the Graphical User Interface (GUI) is given with several screenshots from the various windows of the toolbox. Then, in Chapter 3, a brief theoretical survey is given for the various problems treated. Finally, in Chapter 4, worked examples of the problems presented in Chapter 3, and their solutions using the X-Posse Optimization Toolbox are given.

CHAPTER 2

GRAPHICAL USER-INTERFACE

2.1 General

The Graphical User-Interface (GUI) has been built in MATLAB© programming language, using version R2009b.

The files you need for the installation are:

1. `XPosse_gui1.m`
2. `XPosse_gui1.fig`
3. `tableGUI.m`
4. `tableGUI.fig`

Also, you need the YALMIP optimization toolbox for MATLAB [2], which can be freely downloaded from the following site:

`http://users.isy.liu.se/johanl/yalmip.`

YALMIP is entirely based on m-code, and is thus easy to install. Remove any old version of YALMIP, unzip the file YALMIP.zip and add the following directories to your MATLAB path:

```
/yalmip
/yalmip/extras
/yalmip/demos
```

```
/yalmip/solvers
/yalmip/modules
/yalmip/modules/parametric
/yalmip/modules/moment
/yalmip/modules/global
/yalmip/modules/sos
/yalmip/operators
```

If you have MPT installed, make sure that you delete the YALMIP distribution residing inside MPT and remove the old path definitions.

If you have used YALMIP before, type clear classes or restart MATLAB before using the new version.

The solvers should be installed as described in the solver manuals. Make sure to add required paths.

To test your installation, run the command yalmiptest. For further examples and tests, run code from this Wiki!

If you have problems, please read the FAQ.

YALMIP is primarily developed on a Windows machine using MATLAB 7.12 (2011a). The code should work on any platform, but is developed and thus most extensively tested on Windows. Most parts of YALMIP should in principle work with MATLAB 6.5, but has not been tested (to any larger extent) on these versions. MATLAB 5.2 or earlier versions are definitely not supported.

To start the GUI, simply type in the command window

```
>> XPosse_gui1
```

Then, the window shown in Figure 2.1 appears.

2.2 Selection of nodes

First, the user should define the number of nodes to be used. This number should be written in the textbox under the title *Number of Nodes*. Once the user has typed a number, say N, then a grid of gray filled circles of dimension $N \times N$ appears on the right part of the main window as shown in Figure 2.2.

Then, the user should choose the nodes of interest. To do this, the user should enter the select mode by pressing the radio button *Set Node*. Then, a moving cross appears on the right part of the main window to facilitate selection. The selected nodes are marked as yellow filled circles; see Figure 2.3. Once the maximum number of nodes (*Number of Nodes*) has been reached, the user is not allowed to select more nodes (unless he unselects some and selects some others!).

The user can at any time unselect nodes by entering the unselect mode. This is done, by pressing the radio button *Cancel Node*. Then, a moving cross appears on the right part of the main window to facilitate selection. The unselected nodes are marked back as gray filled circles; see Figure 2.4.

Then, the user can select one or more origin nodes by pressing the radio button *Origin Node*. Then, a moving cross appears on the right part of the main window to facilitate selection. The selected nodes are marked as red filled circles; see Figure 2.5.

Note that, the user shoud choose the origine(s) among the already selected nodes. If he tries to select a gray node, then the error message, shown in Figure 2.6, appears. To unselect one or more origin nodes, the user should click on the selected node and it will be marked back in yellow color.

Finally, the user can select one or more destination nodes by pressing the radio button *Destination Node*. Then, a moving cross appears on the right part of the main window to facilitate selection. The selected nodes are marked as green filled circles; see Figure 2.7.

Again, the user shoud choose the destination(s) among the already selected nodes. If he tries to select a gray node, then an error message appears. To unselect one or more destination nodes, the user should click on the selected node and it will be marked back in yellow color.

During the process of selection/unselection of nodes, origine(s) and destination(s), the coordinates of the current point are shown in the text boxes of *Current Node*.

Once this process has been finished, the user should press the button *Done*. Then, the message shown in Figure 2.8 appears, prompting the user to choose appropriate tables to fill in for the appropriate problem (see below). Also, the nodes are numbered according to the order they have been selected.

2.3 Filling in of tables

The user can select one or more tables to fill in (according to the problem to be solved) by pressing the drop-down menu *Select Table*.

The available tables are (see also Figure 2.9):

a) Cost Table

b) Capacity Table

c) Supply Table

d) Demand Table

By selecting one of the above mentioned tables, a separate window appears; see Figure 2.10. The cells of the table are fully editable.

Each table window has a *File* menu with options *Open*, *Save*, and *Quit*; see Figure 2.11. Using these options, the user can

(i) load an existing table in an ASCII file with extension ".txt".

(ii) save the filled in table in an ASCII file with extension ".txt".

(iii) close the table window and return to the table selection drop-down menu.

Note that, the tables a and b are of dimension $N \times N$, where N is the maximum number of nodes (*Number of Nodes*), while tables c and d are of dimension $1 \times N$.

If a file, containing a table with different dimensions, is loaded an error message is produced. See Figure 2.12.

2.4 Selection of problem

The user can select a problem to be solved by pressing the drop-down menu *Select Problem*.

The available problems are (see also Figure 2.13):

1. Transportation Problem Capacited

2. Transshippment Problem Capacited

3. Shortest Path

4. Maximum Flow

A detailed description of each problem will be given in Chapter 3. Worked examples, and their solutions using GUI, will be given in Chapter 4.

For the solution of the problems, the internal solver *BNB* of YALMIP is used. *BNB* is an implementation of a standard branch & bound algorithm for mixed integer linear/quadratic/second order cone and semidefinite programming solver.

After the successful run of each problem, the results are given in two distinct windows:

a) the *Results* window, containing the value of the Objective function, and

b) the *RESULTS Table* window, containing the Cost table.

See also Figures 2.14 and 2.15.

Moreover, the optimal path is drawn on the right part of the main GUI window; see Figure 2.16.

When calculations have been finished, the user can exit the GUI by pressing the *Exit* button; see Figure 2.17.

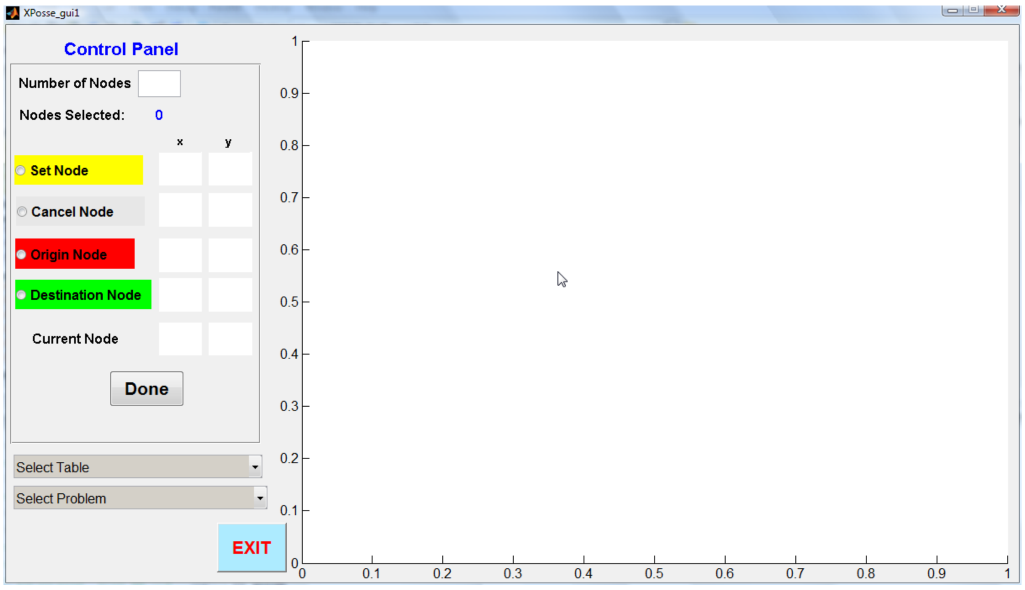


Figure 2.1: Main window of the GUI

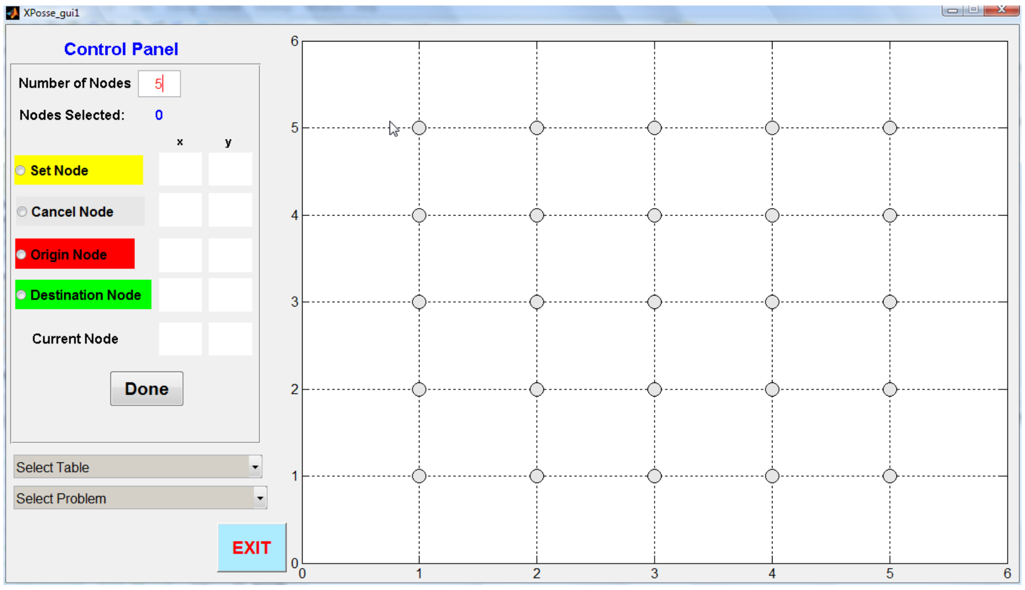


Figure 2.2: Main window of the GUI with grid

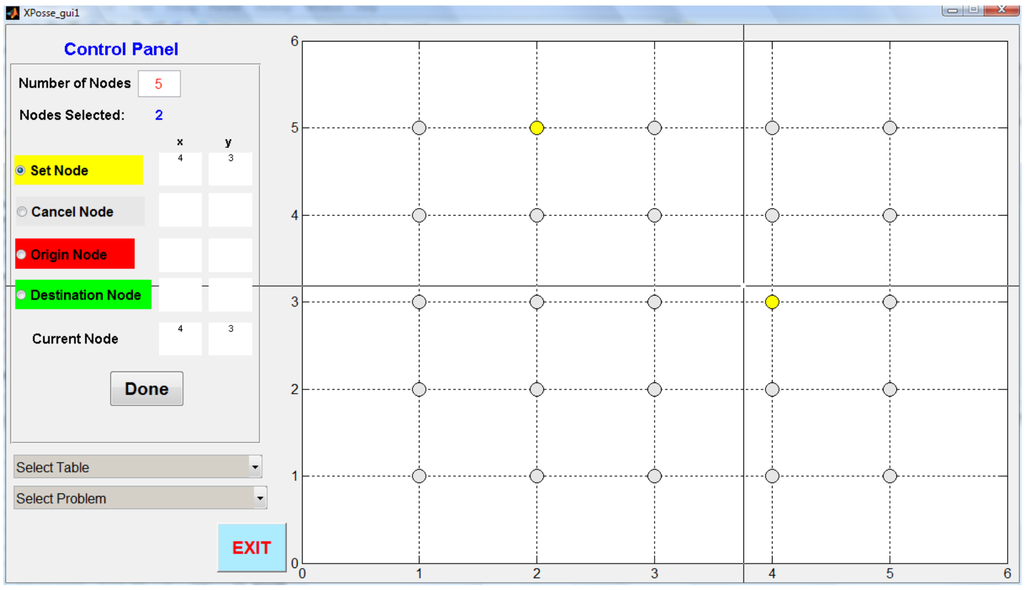


Figure 2.3: Main window of the GUI in select mode

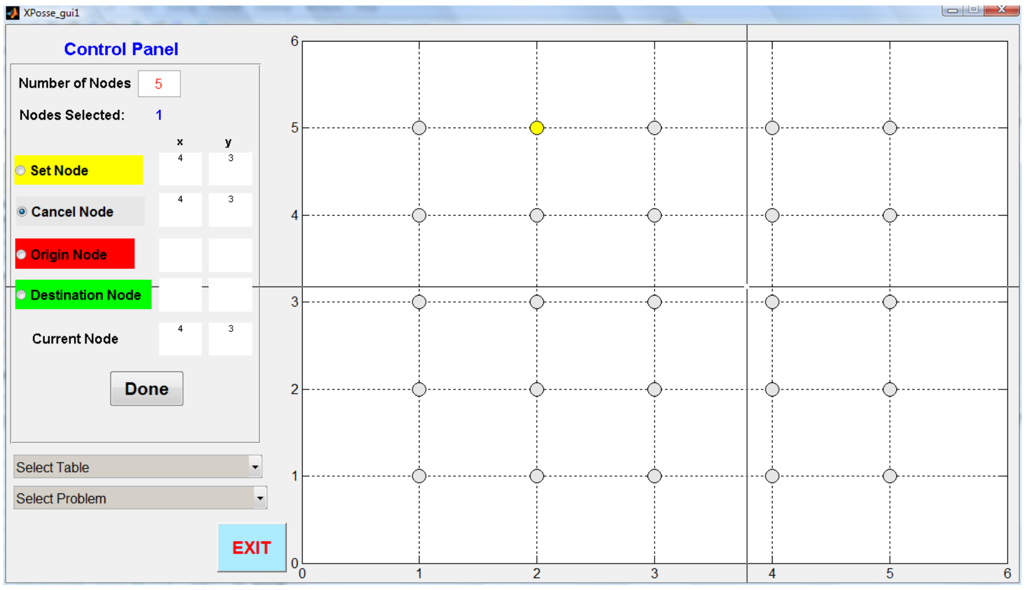
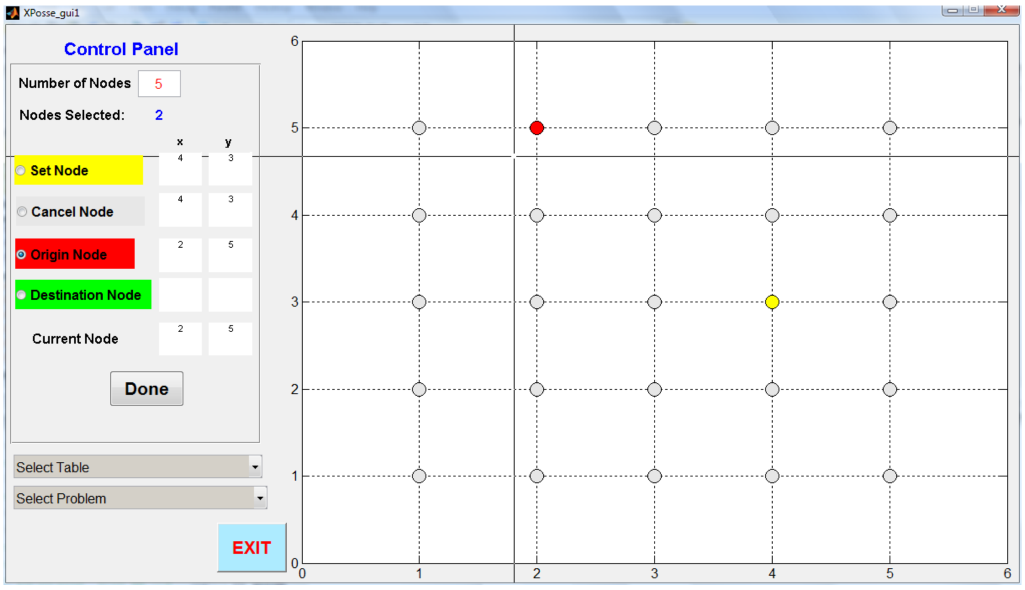


Figure 2.4: Main window of the GUI in unselect mode

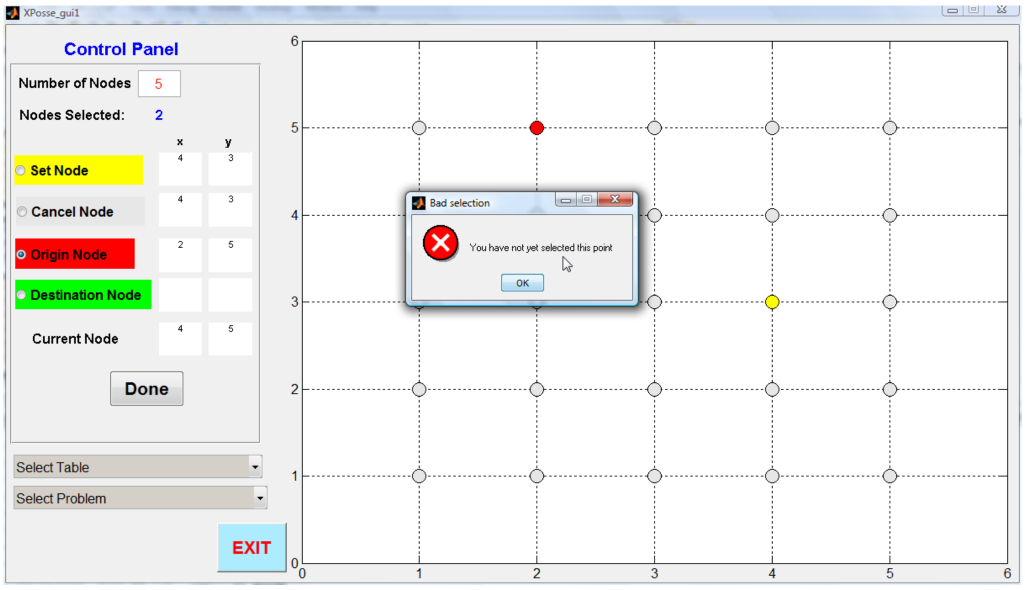


Number of Nodes 5

Nodes Selected: 2

	x	y
Set Node	4	3
Cancel Node	4	3
Origin Node	2	5
Destination Node		
Current Node	2	5

Figure 2.5: Main window of the GUI in “origine” mode



Number of Nodes 5

Nodes Selected: 2

	x	y
Set Node	4	3
Cancel Node	4	3
Origin Node	2	5
Destination Node		
Current Node	4	5

You have not yet selected this point

Figure 2.6: Main window of the GUI in “origine” mode. Error message

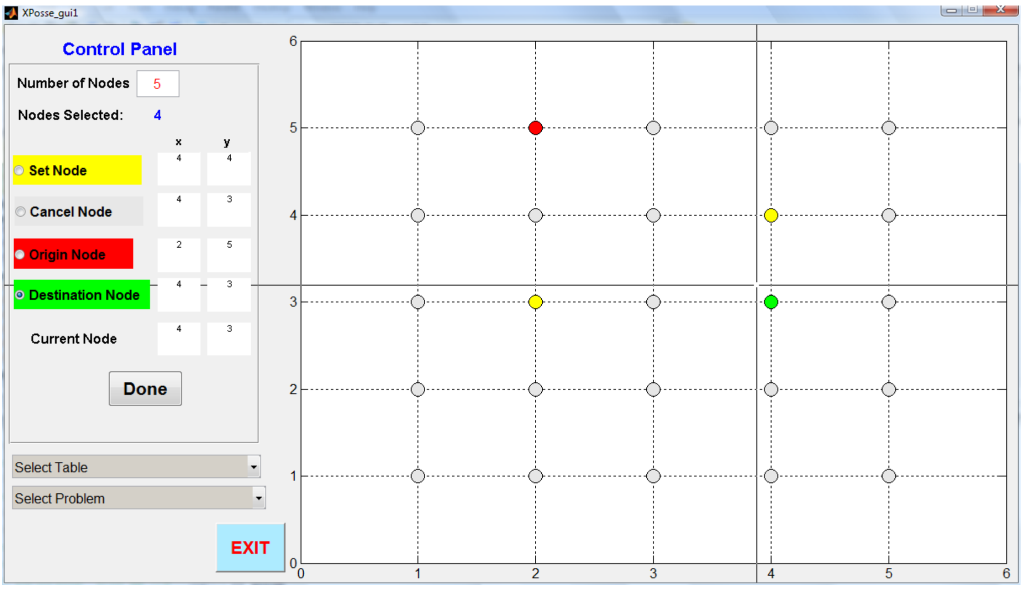
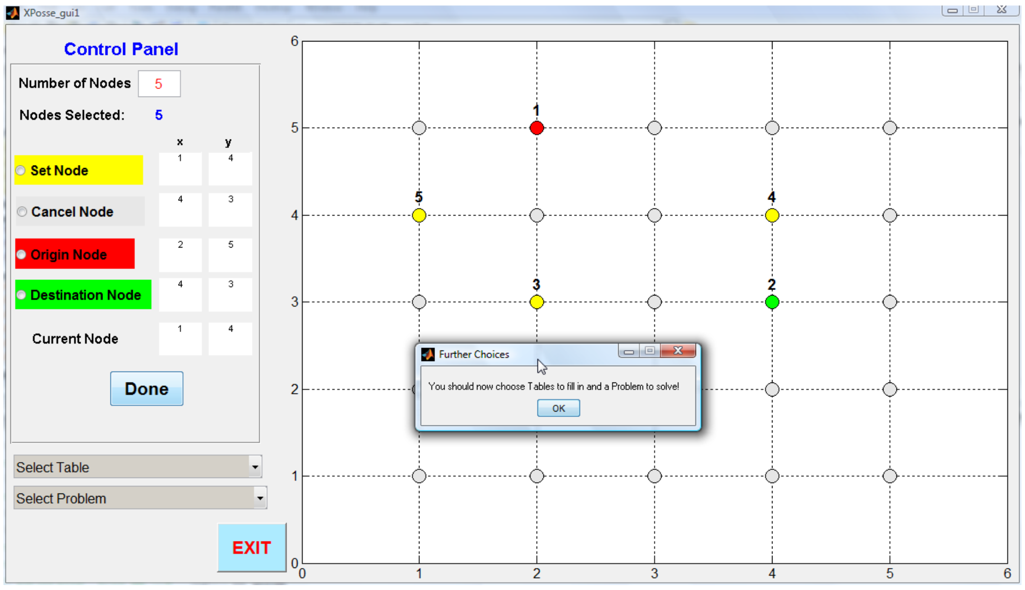


Figure 2.7: Main window of the GUI in “destination” mode



You should now choose Tables to fill in and a Problem to solve!

Figure 2.8: Main window of the GUI with button *Done* pressed

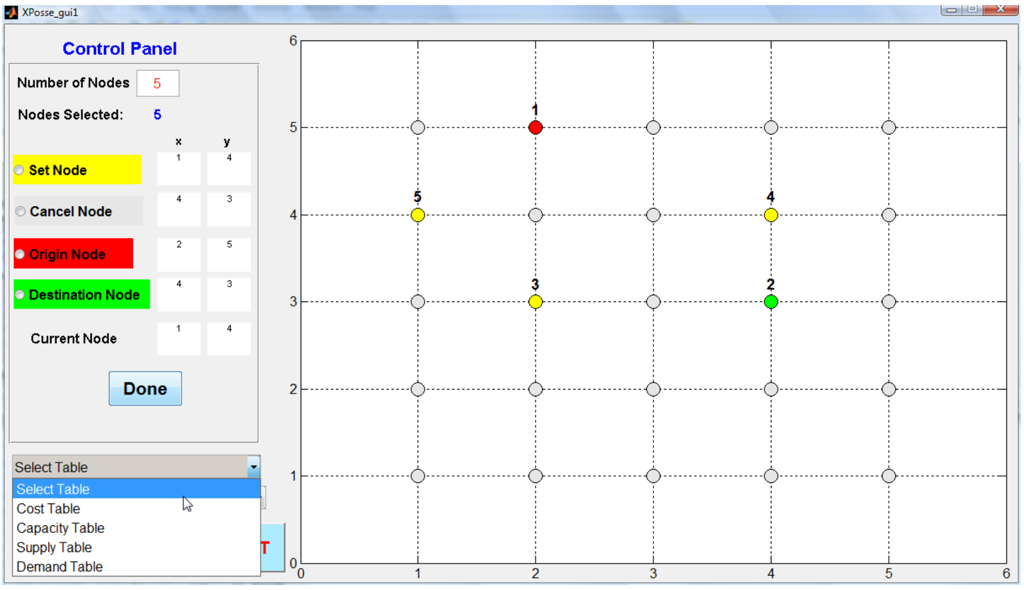
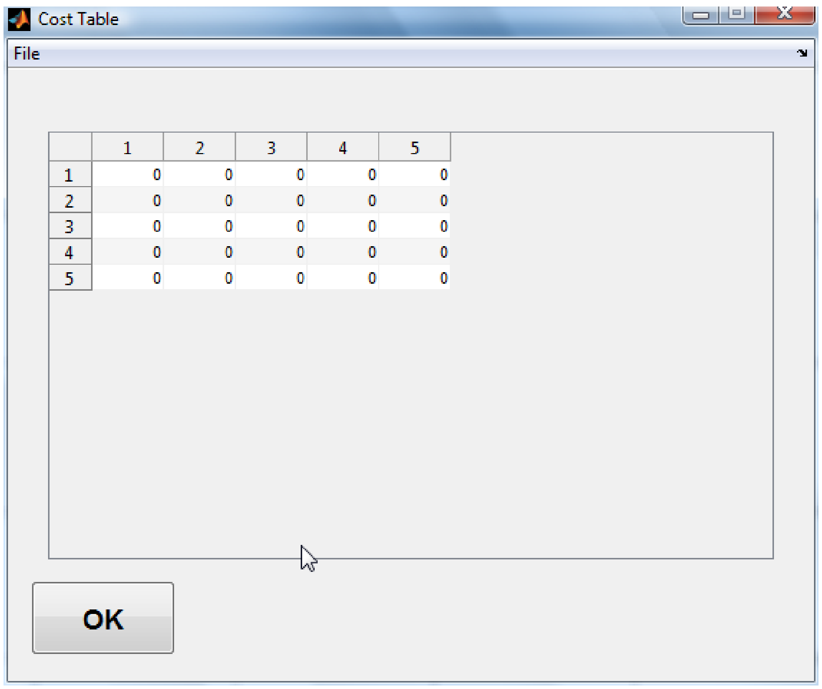
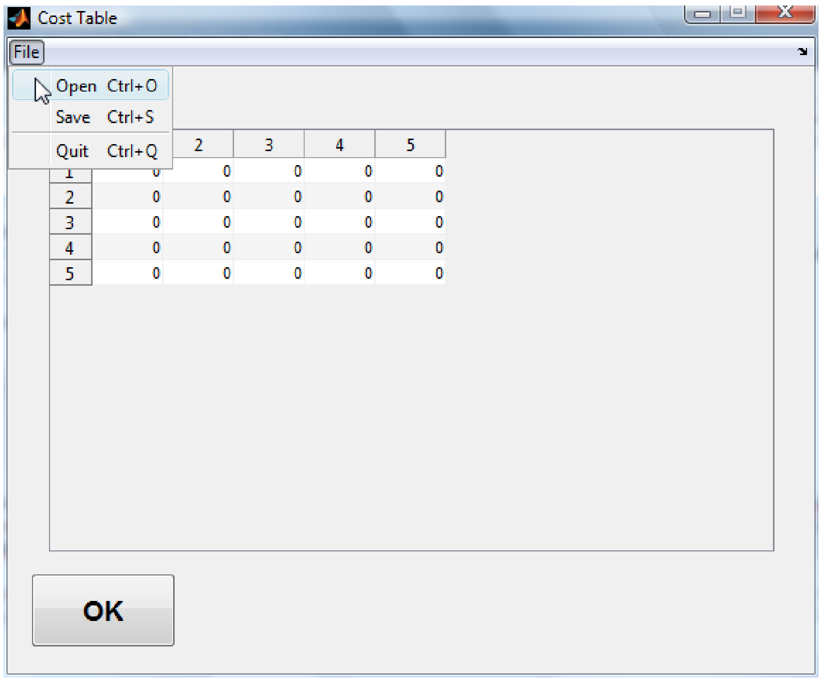


Figure 2.9: Main window of the GUI with *Select Table* pressed



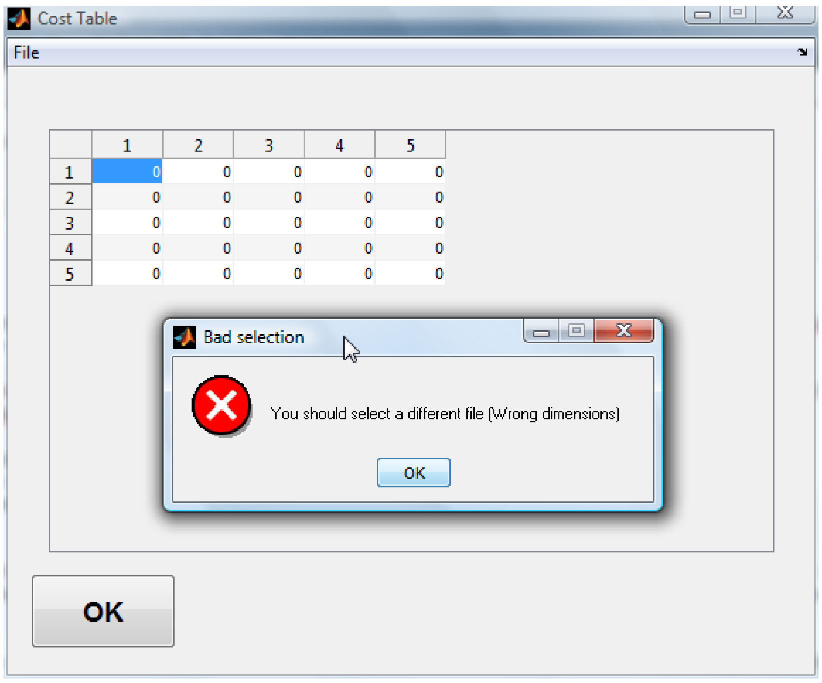
	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

Figure 2.10: Table window



		2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

Figure 2.11: Table window options



	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

You should select a different file (Wrong dimensions)

Figure 2.12: Table window. Error message

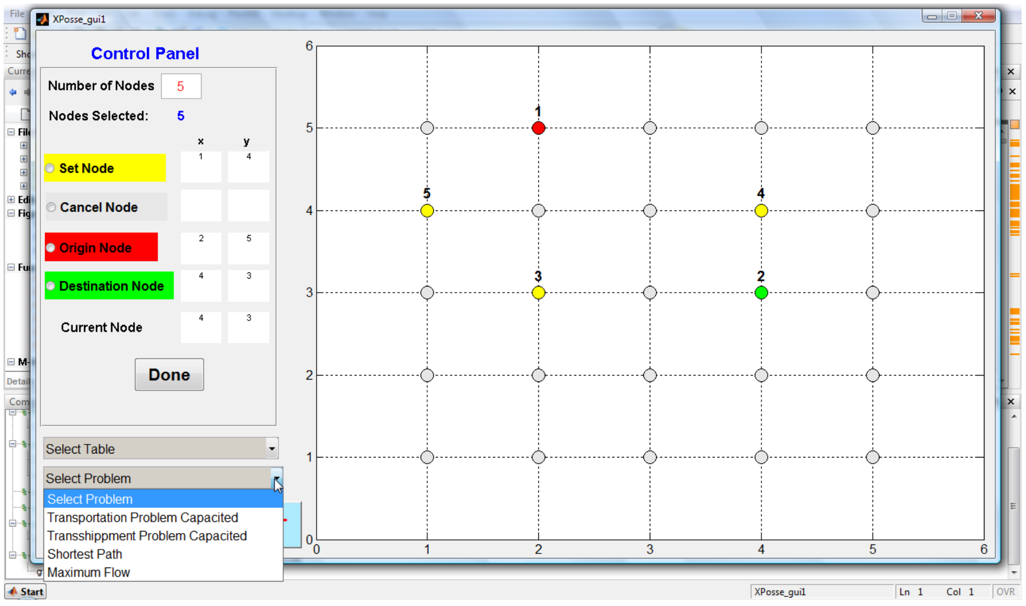
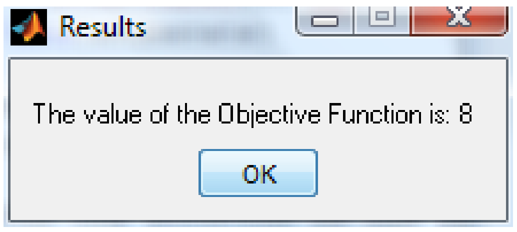
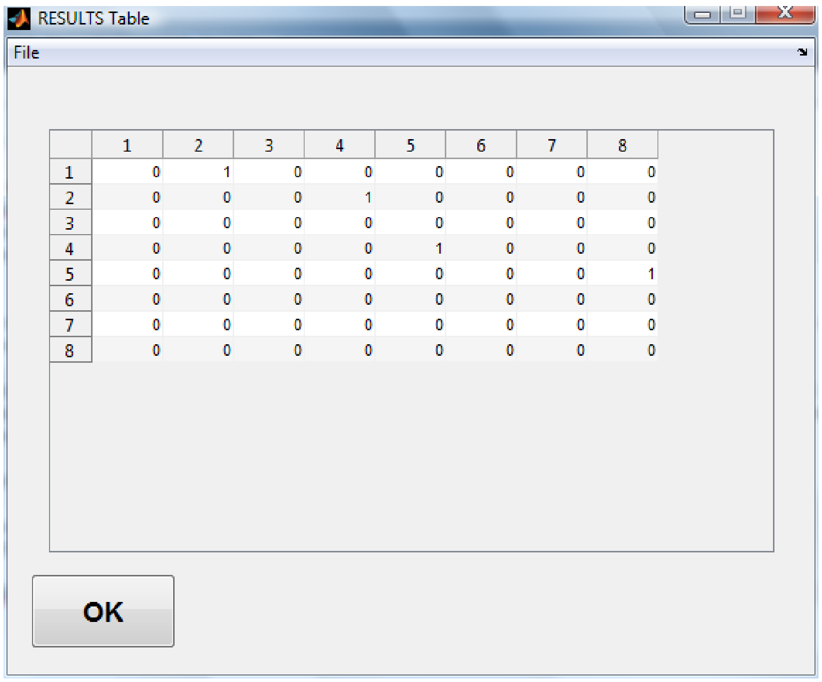


Figure 2.13: Main window of the GUI with *Select Problem* pressed



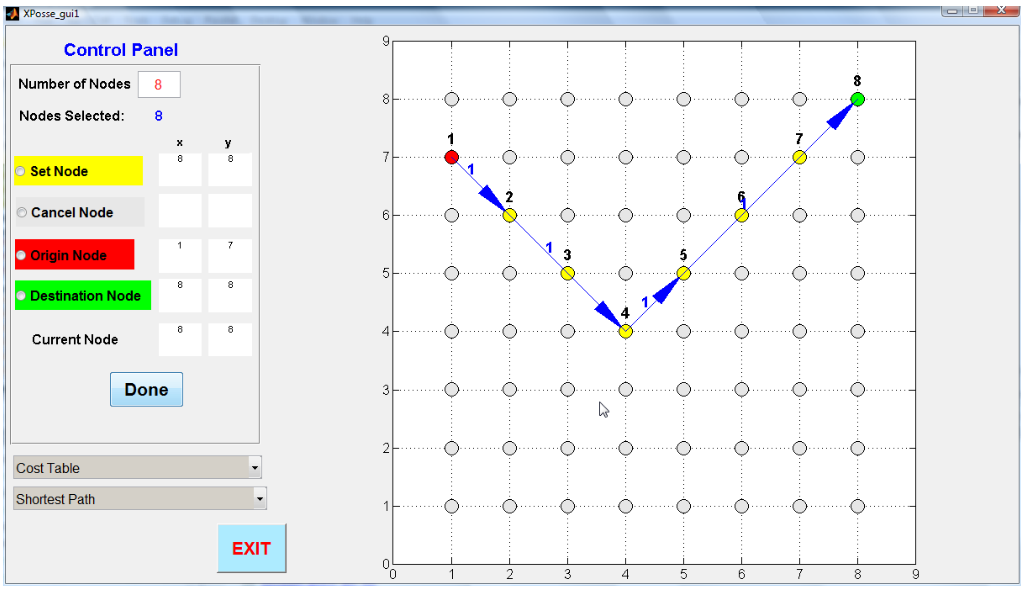
The value of the Objective Function is: 8

Figure 2.14: *Results* window



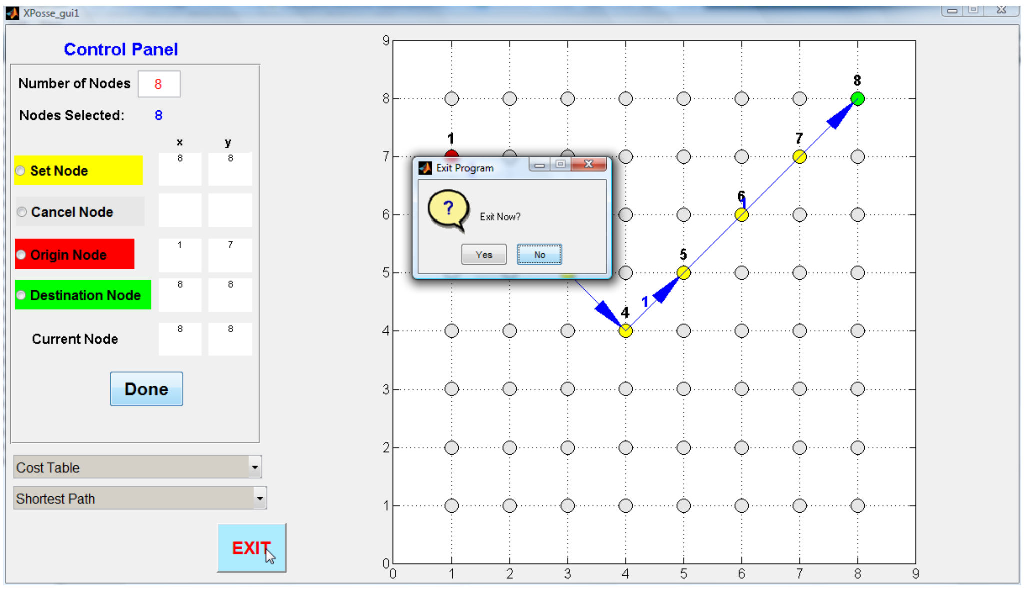
	1	2	3	4	5	6	7	8
1	0	1	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Figure 2.15: *RESULTS Table* window



	x	y
Number of Nodes	8	
Nodes Selected:	8	
Set Node	8	8
Cancel Node		
Origin Node	1	7
Destination Node	8	8
Current Node	8	8

Figure 2.16: Main window of the GUI with Optimal Solution drawn


XPosse_gui1

Control Panel

Number of Nodes 8

Nodes Selected: 8

	x	y
Set Node	8	8
Cancel Node		
Origin Node	1	7
Destination Node	8	8
Current Node	8	8

Done

Cost Table

Shortest Path

EXIT

Exit Program

Exit Now?

Yes No

Figure 2.17: Main window of the GUI with *Exit* button pressed

CHAPTER 3

THEORY

In the present chapter, a detailed description of the problems treated by the X-Posse Optimization Toolbox will be given.

For further reading, the interested user is advised to look into any standard reference book; see, e.g., [3, 4].

Before starting with the description of the problems, let us first give some definitions. In all problems mentioned below, a network of points, usually called nodes, is used. The transfer of goods from point i to the point j has a cost c_{ij}. Also, the distance between point i and point j is called arc (i, j). Note that, the cost $c_{ii} = 0$ for any point i.

3.1 Transportation Problem Capacited

A transportation problem can be specified as follows.

Let us assume that we have m number of supply points (origins) and n number of demand points (destinations). Usually, supply and demand points refer to as “sources” and “sinks”.

Assume further that each supply point i can supply at most S_i units and each demand point j must receive at least D_j units of the shipped good.

Let x_{ij} be the number of units shipped from supply point i to demand point j at a (variable) cost c_{ij}. Note that, in the capacited version of the problem, x_{ij} should not exceed capacity u_{ij} of the arc (from i to j).

In order to minimize the cost for transporting goods from the m origins to the n destinations, without exceeding the capacity u_{ij} of each arc (i, j), we have to solve the following Mixed-Integer Linear Programming Optimization Problem:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{3.1}$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = S_i, \quad i = 1, 2, \ldots, m \tag{3.2}$$

$$\sum_{i=1}^{m} x_{ij} = D_j, \quad j = 1, 2, \ldots, n \tag{3.3}$$

$$0 \le x_{ij} \le u_{ij}, \quad \forall i, j \tag{3.4}$$

In this point, we make the following observations:

(i) The total number of variables x_{ij} is $m \times n$.

(ii) The total number of constraints is $2(m+n)$.

(iii) The total supply is assumed to be balanced by the total demand, i.e.,

$$\sum_{i=1}^{m} S_i = \sum_{j=1}^{n} D_j \tag{3.5}$$

In a different case, the GUI takes internally care to add a dummy demand point (if supply is greater than demand) or a dummy supply point (if supply is less than demand) with zero cost.

3.2 Transshippment Problem Capacited

The transshipment problem is very similar to the transportation problem. In the previous section, we defined supply points as points, from which goods can only be sent, and demand points as points, to which goods can only be received. On the other hand, transshipment points are defined as points from which goods can both be sent and received to and from other points.

Thus, keeping the notation of the previous section, and in order to minimize the cost for transporting goods from the m origins to the n destinations, without exceeding the capacity u_{ij} of each arc (i, j), we have to solve the following Mixed-Integer Linear Programming Optimization Problem:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{3.6}$$

s.t.

$$\sum_{j=1}^{n} x_{ij} \le S_i, \quad i = 1, 2, \ldots, m \tag{3.7}$$

$$\sum_{k} x_{ik} - \sum_{k} x_{kj} = 0, \quad i, j = \text{transshipment points} \tag{3.8}$$

$$\sum_{i=1}^{m} x_{ij} = D_j, \quad j = 1, 2, \ldots, n \tag{3.9}$$

$$0 \le x_{ij} \le u_{ij} \quad \forall i, j \tag{3.10}$$

Note that, if there is an outflow from a demand point to another demand point, this should be taken into account in Equation (3.9).

3.3 Shortest Path

In this section, we assume that each arc in the network of points has a length associated with it. Suppose we start at a particular node (say, node 1). The problem of finding the shortest path (path of minimum length) from node 1 to any other node in the network is called a Shortest Path problem.

Finding the shortest path between node i and node j in a network may be viewed as a transshipment problem. Simply try to minimize the cost of sending one unit from node i to node j (with all other nodes in the network being transshipment points), where the cost of sending one unit from node k to node k' is the length of arc (k, k') if such an arc exists and is M (a large positive number) if such an arc does not exist. As in previous sections, the cost of shipping one unit from a node to itself is zero.

Thus, keeping the notation of the previous sections, we have to solve the following Mixed-Integer Linear Programming Optimization Problem:

$$\min \sum_{i} \sum_{j \ne i} c_{ij} x_{ij} \tag{3.11}$$

s.t.

$$\sum_{k \ne j} x_{jk} - \sum_{k \ne j} x_{kj} = \begin{cases} 1, & \text{if } j \text{ is origin} \\ 0, & \text{if } j \text{ is not origin/destination} \\ -1, & \text{if } j \text{ is destination} \end{cases} \tag{3.12}$$

$$0 \le x_{ij} \quad \forall i, j \tag{3.13}$$

3.4 Maximum Flow

Many situations can be modelled by a network in which the arcs may be thought of as having a capacity that limits the quantity of a product that may be shipped through the arc.

In these situations, it is often desired to transport the maximum amount of flow from an origin point (called the source) to a destination point (called the sink).

Such problems are called Maximum Flow problems.

Denoting by v this amount of flow, we have to solve the following Mixed-Integer Linear Programming Optimization Problem:

$$\max v \tag{3.14}$$

s.t.

$$\sum_{k\neq j} x_{jk} - \sum_{k\neq j} x_{kj} = \begin{cases} v, & \text{if } j \text{ is source} \\ 0, & \text{if } j \text{ is not source/sink} \\ -v, & \text{if } j \text{ is sink} \end{cases} \tag{3.15}$$

CHAPTER 4

EXAMPLES

In the present chapter, worked examples of the problems presented in Chapter 3, and their solutions using the X-Posse Optimization Toolbox are given.

4.1 Transportation Problem Capacited

Let a network of 7 points, where goods can be transported from the first 3 points (supply points) to the next 4 points (demand points); see Figure 4.1.

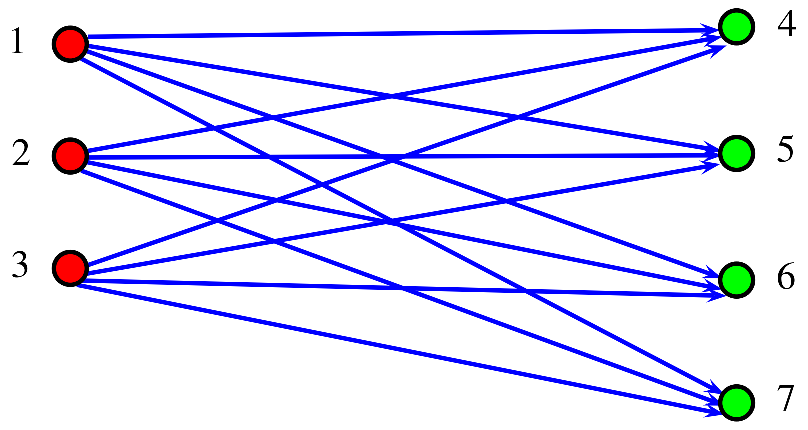


Figure 4.1: Graphical representation of the network for the Transportation Problem

Let us further suppose that the cost, supply, and demand tables have the following values:

(i) Cost table

$$c = \begin{bmatrix} 0 & 0 & 0 & 120 & 130 & 41 & 62 \\ 0 & 0 & 0 & 61 & 40 & 100 & 110 \\ 0 & 0 & 0 & 102.5 & 90 & 122 & 42 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.1}$$

(ii) Supply table

$$S = \begin{bmatrix} 500 & 700 & 800 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.2}$$

(iii) Demand table

$$D = \begin{bmatrix} 0 & 0 & 0 & 400 & 900 & 200 & 500 \end{bmatrix} \tag{4.3}$$

Moreover, we assume that the capacity of the supply points is very high (practically unlimited). In our example, we have put a very high number for the capacity of each point. Thus, the Capacity table is as follows:

$$u = \begin{bmatrix} 0 & 0 & 0 & 100000 & 100000 & 100000 & 100000 \\ 0 & 0 & 0 & 100000 & 100000 & 100000 & 100000 \\ 0 & 0 & 0 & 100000 & 100000 & 100000 & 100000 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.4}$$

First, you type 7 to the textbox *Number of Nodes*. Then, a grid appears on the right of the main window. Then, you select origin points by selecting radio button *Origin Node* and destination points by selecting *Destination Node*. Try to remember the order of selection, because the first three should be defined as origin points and the last four as destination points. See also Figure 4.2.

Then, you have to fill in the tables by selecting them from the drop-down menu *Select Table*. Either you can type the numbers or you can load the files. The tables are already saved in ASCII files and can be directly loaded to the GUI from the current directory. The filenames are:

```
CTPPcapacity.txt
CTPPcost.txt
CTPPdemand.txt
CTPPsupply.txt
```

Finally, from the drop-down menu *Select Problem* you select *Transportation Problem Capacited*.

The results obtained are:

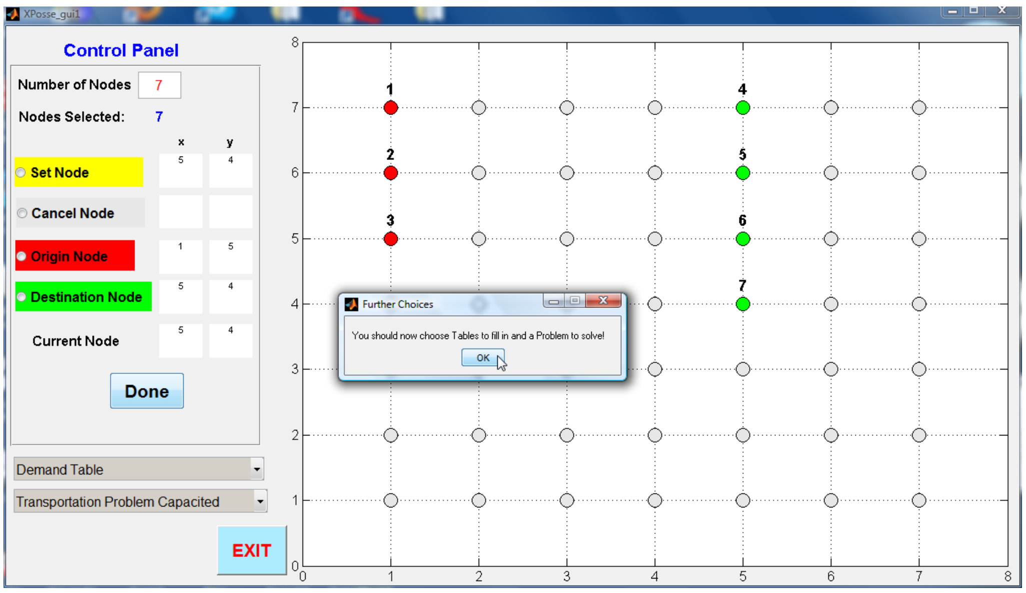
a) Results (minimum cost) = 121450

b) RESULTS Table (optimum quantity)

$$oq = \begin{bmatrix} 0 & 0 & 0 & 300 & 0 & 200 & 0 \\ 0 & 0 & 0 & 0 & 700 & 0 & 0 \\ 0 & 0 & 0 & 100 & 200 & 0 & 500 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.5}$$

and they are given in different windows. See also Figures 4.3 and 4.4.

Also, on the right part of the main window, supply points have been connected to the demand points by arrows indicating the optimum path. The values above the arrows come from the resulting table. See also Figure 4.5.



Number of Nodes 7

Nodes Selected: 7

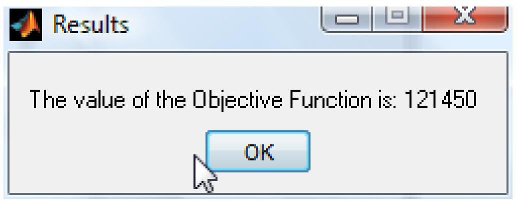
	x	y
Set Node	5	4
Cancel Node		
Origin Node	1	5
Destination Node	5	4
Current Node	5	4

Demand Table

Transportation Problem Capacited

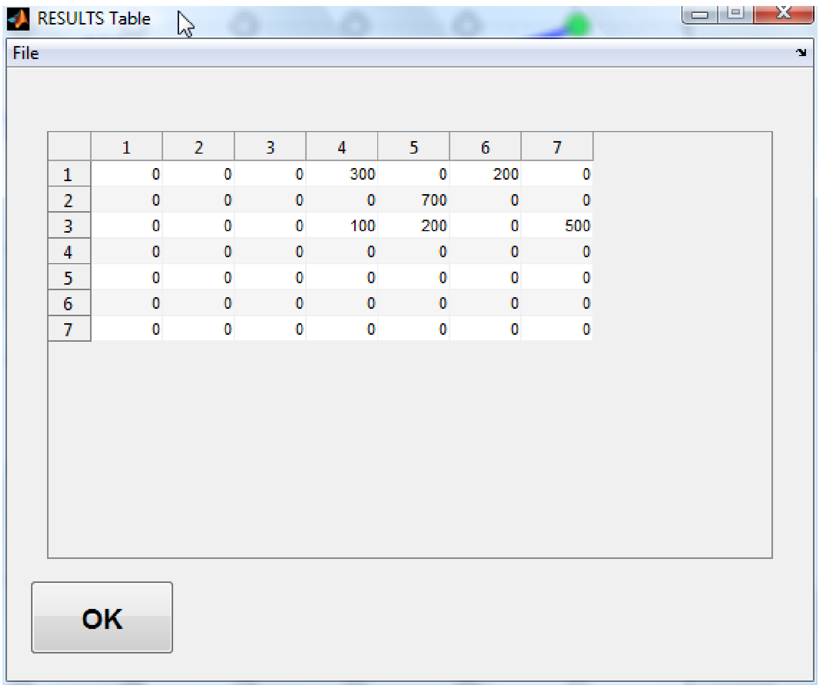
You should now choose Tables to fill in and a Problem to solve!

Figure 4.2: Transportation Problem. Main window after selection of points



The value of the Objective Function is: 121450

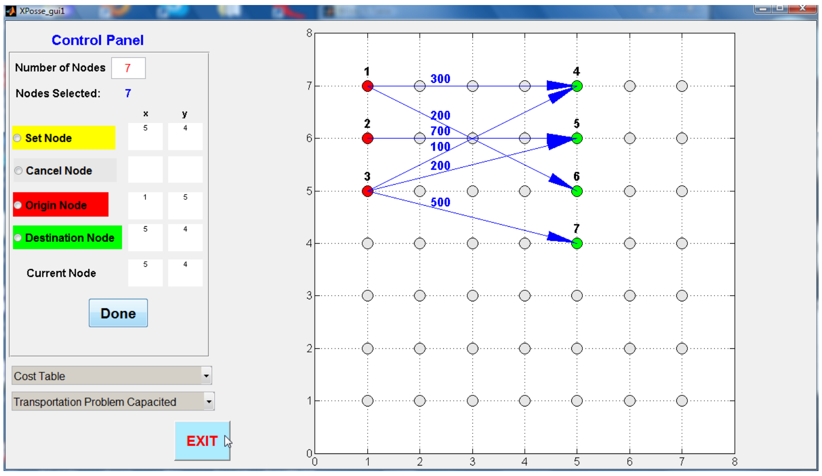
Figure 4.3: Transportation Problem. *Results* window


RESULTS Table

File

	1	2	3	4	5	6	7
1	0	0	0	300	0	200	0
2	0	0	0	0	700	0	0
3	0	0	0	100	200	0	500
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0

Figure 4.4: Transportation Problem. *RESULTS Table* window


Control Panel

Number of Nodes 7

Nodes Selected: 7

	x	y
Set Node	5	4
Cancel Node		
Origin Node	1	5
Destination Node	5	4
Current Node	5	4

Cost Table

Transportation Problem Capacited

Figure 4.5: Transportation Problem. Optimum path

4.2 Transshippment Problem Capacited

Let a network of 5 points, where now supply point is only the first (No. 1) and demand points are the third and the fourth (No. 3 and 4). The remaining two points (No. 2 and 5) are transshipment points; see Figure 4.6.

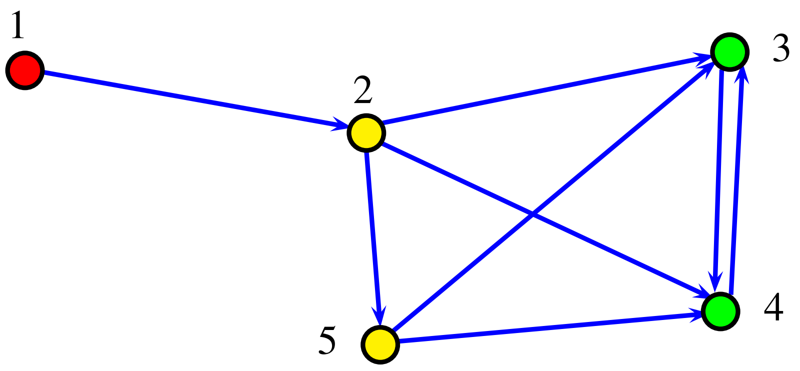


Figure 4.6: Graphical representation of the network for the Transshippment Problem

Let us further suppose that the cost, supply, and demand tables have the following values:

(i) Cost table

$$c = \begin{bmatrix} 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 45 & 50 & 20 \\ 0 & 0 & 0 & 60 & 0 \\ 0 & 0 & 85 & 0 & 0 \\ 0 & 0 & 10 & 55 & 0 \end{bmatrix} \tag{4.6}$$

(ii) Supply table

$$S = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.7}$$

(iii) Demand table

$$D = \begin{bmatrix} 0 & 0 & 3 & 7 & 0 \end{bmatrix} \tag{4.8}$$

Moreover, we assume that the Capacity table is as follows:

$$u = \begin{bmatrix} 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 4 & 3 & 3 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 3 & 5 & 0 \end{bmatrix} \tag{4.9}$$

First, you type 5 to the textbox *Number of Nodes.* Then, a grid appears on the right of the main window. Then, you select origin point by selecting radio button *Origin Node* and destination points by selecting *Destination Node.* Try to remember the order of selection, because the first should be defined as origin point, and the third and fourth as destination points. See also Figure 4.7.

Then, you have to fill in the tables by selecting them from the drop-down menu *Select Table.* Either you can type the numbers or you can load the files. The tables are already saved in ASCII files and can be directly loaded to the GUI from the current directory. The filenames are:

```
CTSPcapacity.txt
CTSPcost.txt
CTSPdemand.txt
CTSPsupply.txt
```

Finally, from the drop-down menu *Select Problem* you select *Transshipment Problem Capacited.*

The results obtained are:

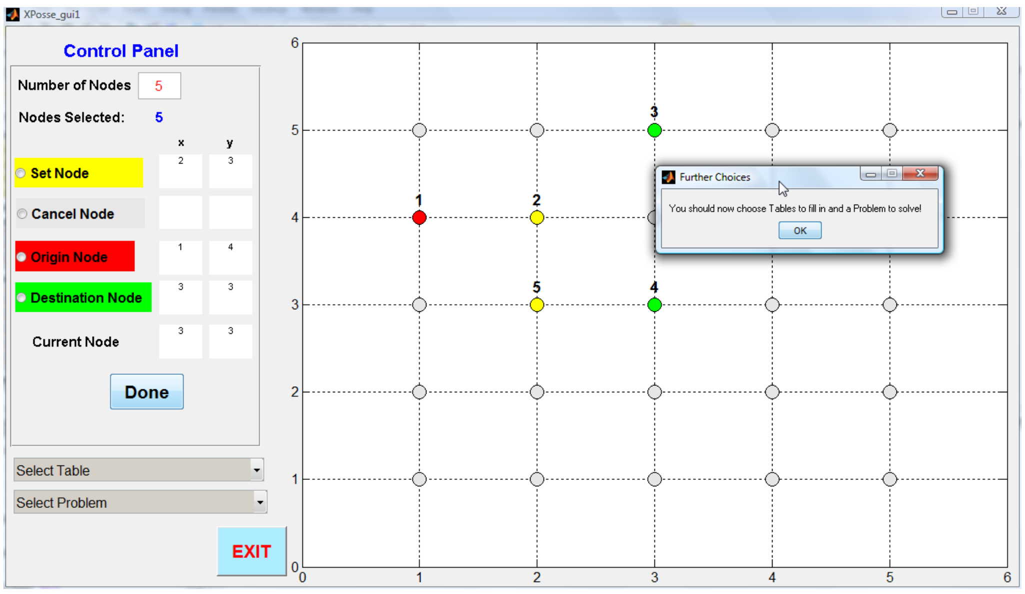
a) Results (minimum cost) = 1615

b) RESULTS Table (optimum quantity)

$$oq = \begin{bmatrix} 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 4 & 3 & 3 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \end{bmatrix} \tag{4.10}$$

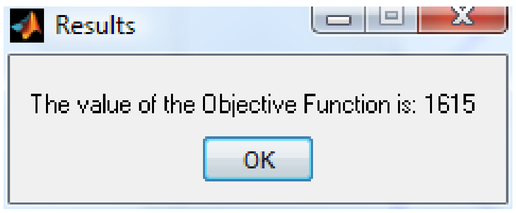
and they are given in different windows. See also Figures 4.8 and 4.9.

Also, on the right part of the main window, all points are interconnected by arrows indicating the optimum path. The values above the arrows come from the resulting table. See also Figure 4.10.



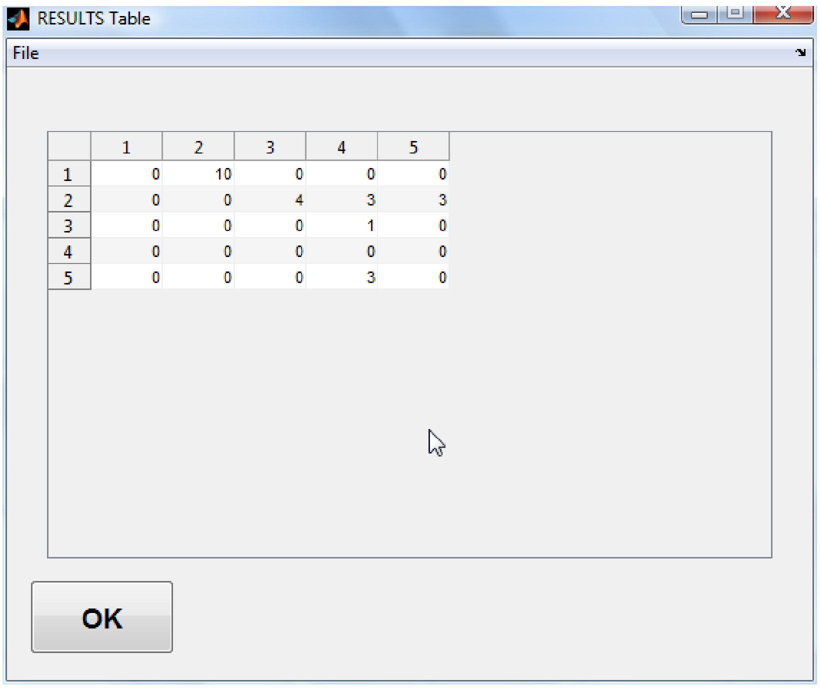
You should now choose Tables to fill in and a Problem to solve!

Figure 4.7: Transshipment Problem. Main window after selection of points



The value of the Objective Function is: 1615

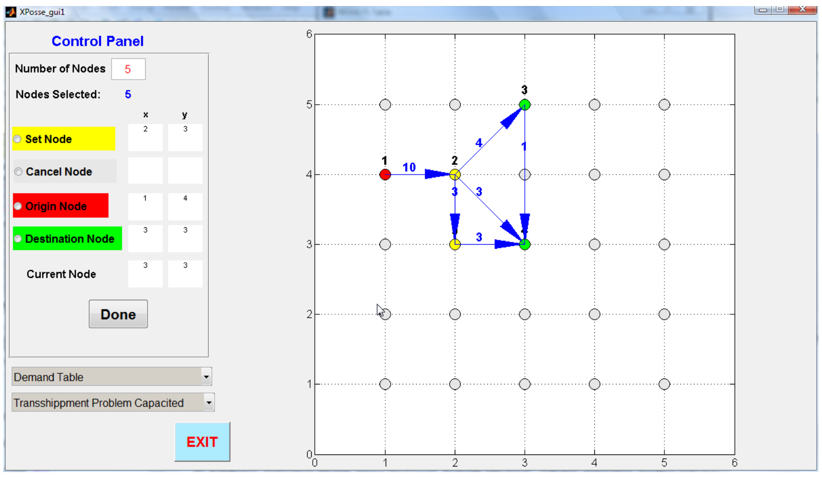
Figure 4.8: Transshipment Problem. *Results* window


RESULTS Table

File

	1	2	3	4	5
1	0	10	0	0	0
2	0	0	4	3	3
3	0	0	0	1	0
4	0	0	0	0	0
5	0	0	0	3	0

Figure 4.9: Transshipment Problem. *RESULTS Table* window


XPosse_gui1

Control Panel

	x	y
Number of Nodes: 5		
Nodes Selected: 5		
Set Node	2	3
Cancel Node		
Origin Node	1	4
Destination Node	3	3
Current Node	3	3

Demand Table

Transshippment Problem Capacited

Figure 4.10: Transshipment Problem. Optimum path

4.3 Shortest Path

Let a network of 8 points, and we want to find the shortest path from point No. 1 to point No. 6; see Figure 4.11.

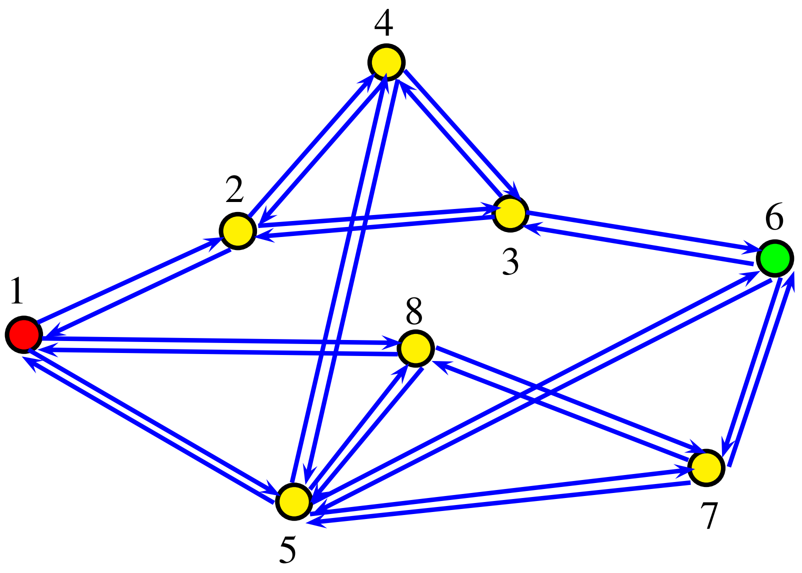


Figure 4.11: Graphical representation of the network for the Shortest Path problem

Let us further suppose that the cost table is the following:
Cost table

$$c = \begin{bmatrix} 0 & 4 & 0 & 0 & 7 & 0 & 0 & 8 \\ 4 & 0 & 6 & 1 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 1 & 0 & 2 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 7 & 0 & 0 & 1 & 0 & 3 & 3 & 2 \\ 0 & 0 & 2 & 0 & 3 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 3 & 3 & 0 & 1 \\ 8 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \end{bmatrix} \tag{4.11}$$

Let us remind that, in the present problem, the cost from point i to point j is the Euclidean distance between the two points.

First, you type 8 to the textbox *Number of Nodes*. Then, a grid appears on the right of the main window. Then, you select origin point by selecting radio button *Origin Node* and destination point by selecting *Destination Node*. Try to remember the order of selection, because the first should be defined as origin point, and the sixth as destination point. See also Figure 4.12.

Then, you have to fill in the cost table by selecting it from the drop-down menu *Select Table*. Either you can type the numbers or you can load the file. The table is already saved in ASCII file and can be directly loaded to the GUI from the current directory. The filename is:

```
SPcost.txt
```

Finally, from the drop-down menu *Select Problem* you select *Shortest Path*. The results obtained are:

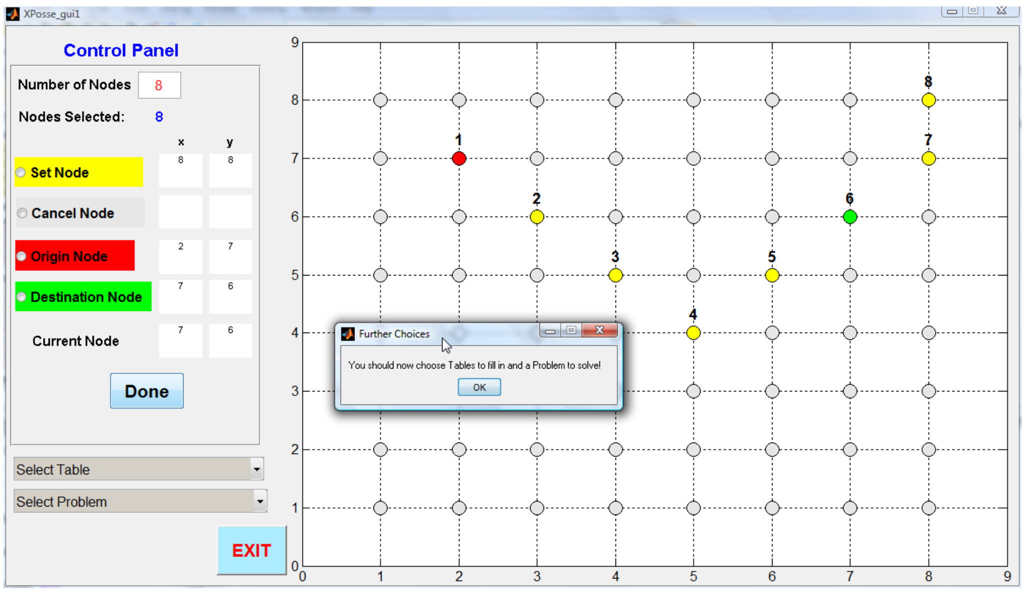
a) Results (minimum distance) = 8

b) RESULTS Table (optimum path, 1=connection, 0=no connection between 2 points)

$$op = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.12}$$

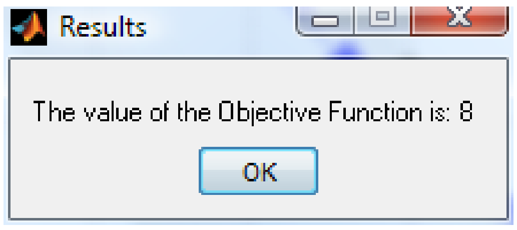
and they are given in different windows. See also Figures 4.13 and 4.14.

Also, on the right part of the main window, the points consisting the optimum path are interconnected by arrows. The values above the arrows come from the resulting table. In the present problem, 1 means connection between 2 points. See also Figure 4.15.



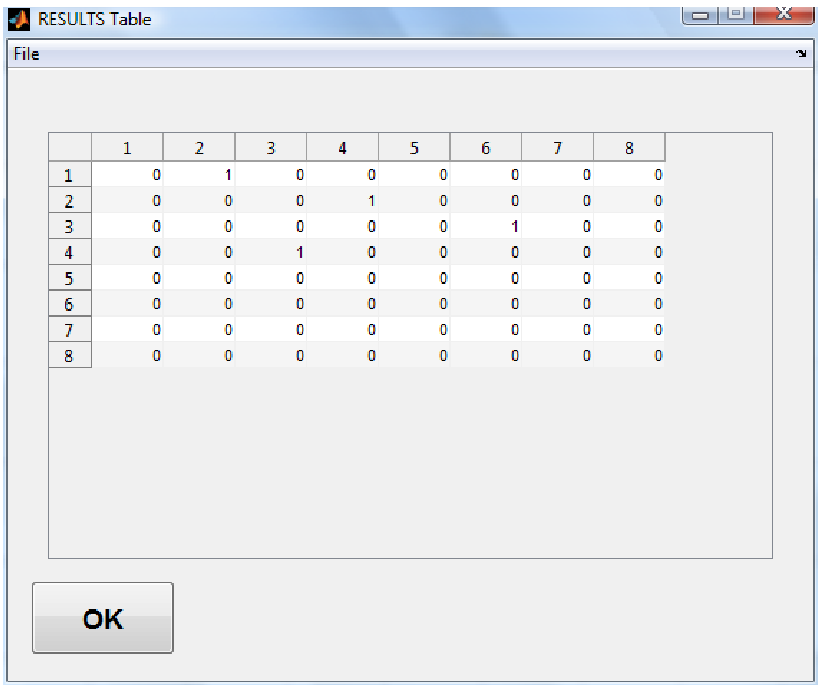
You should now choose Tables to fill in and a Problem to solve!

Figure 4.12: Shortest Path. Main window after selection of points



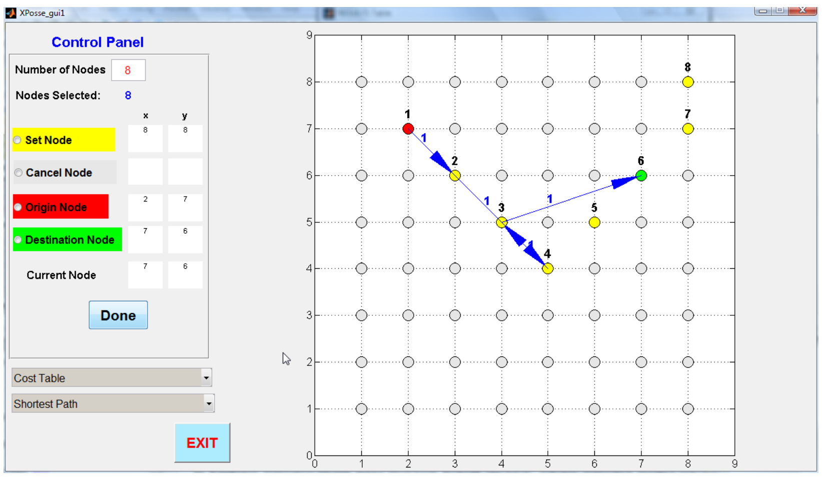
The value of the Objective Function is: 8

Figure 4.13: Shortest Path. *Results* window



	1	2	3	4	5	6	7	8
1	0	1	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0
3	0	0	0	0	0	1	0	0
4	0	0	1	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Figure 4.14: Shortest Path. *RESULTS Table* window



Number of Nodes 8
Nodes Selected: 8

	x	y
Set Node	8	8
Cancel Node		
Origin Node	2	7
Destination Node	7	6
Current Node	7	6

Figure 4.15: Shortest Path. Optimum path

4.4 Maximum Flow

Let a network of 6 points, and we want to maximize the flow from point No. 1 to point No. 6; see Figure 4.16.

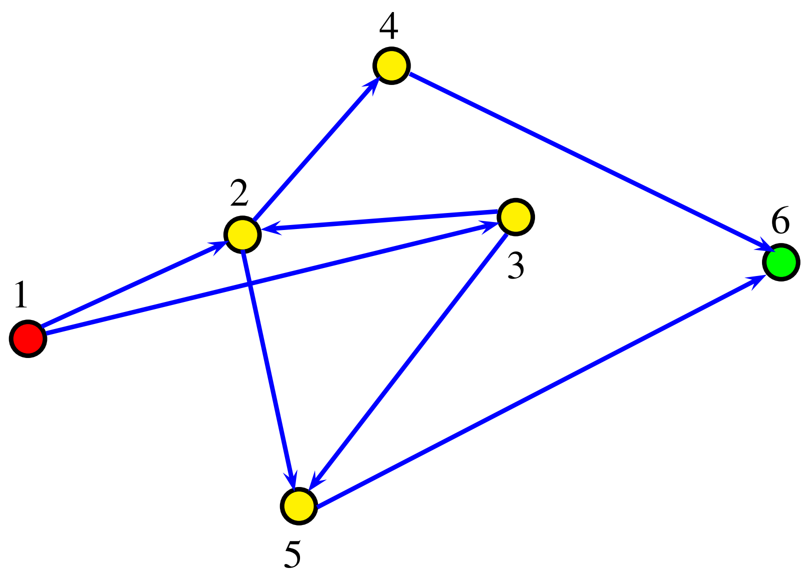


Figure 4.16: Graphical representation of the network for the Maximum Flow problem

Let us further suppose that the capacity table is the following:
Capacity table

$$u = \begin{bmatrix} 0 & 4 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 2 & 0 \\ 0 & 3 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.13}$$

First, you type 6 to the textbox *Number of Nodes*. Then, a grid appears on the right of the main window. Then, you select origin point by selecting radio button *Origin Node* and destination point by selecting *Destination Node*. Try to remember the order of selection, because the first should be defined as origin point, and the sixth as destination point. See also Figure 4.17.

Then, you have to fill in the capacity table by selecting it from the drop-down menu *Select Table*. Either you can type the numbers or you can load the file. The table is already saved in ASCII file and can be directly loaded to the GUI from the current directory. The filename is:

```
MFcapacity.txt
```

Finally, from the drop-down menu *Select Problem* you select *Maximum Flow*.

The results obtained are:

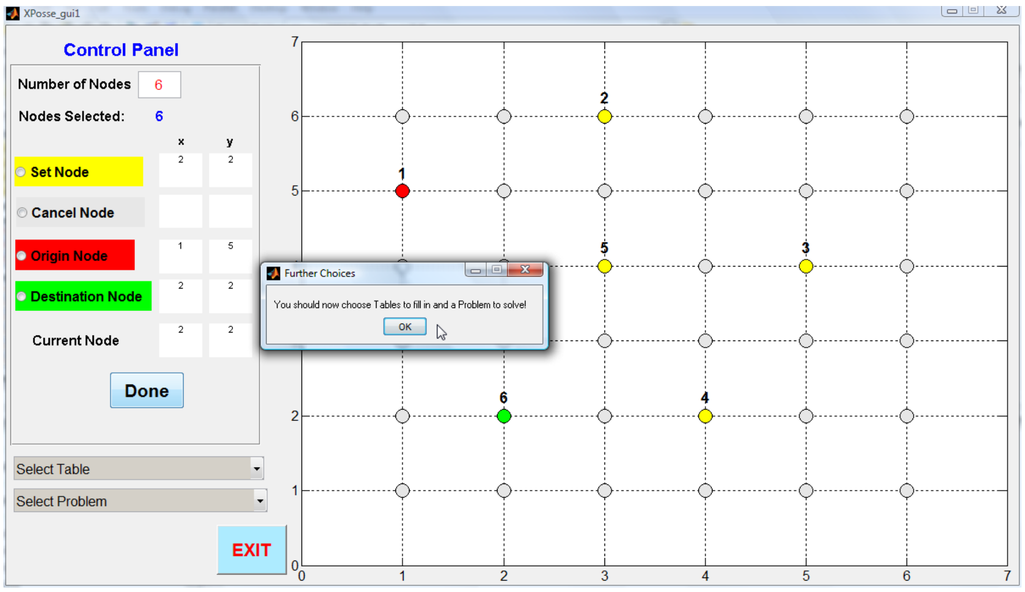
a) Results (maximum flow) = 8

b) RESULTS Table (optimum path)

$$op = \begin{bmatrix} 0 & 4 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 2 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.14}$$

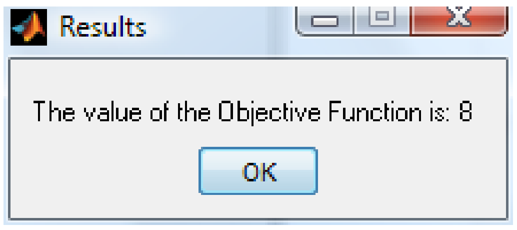
and they are given in different windows. See also Figures 4.18 and 4.19.

Also, on the right part of the main window, the points consisting the optimum path are interconnected by arrows. The values above the arrows come from the resulting table. See also Figure 4.20.



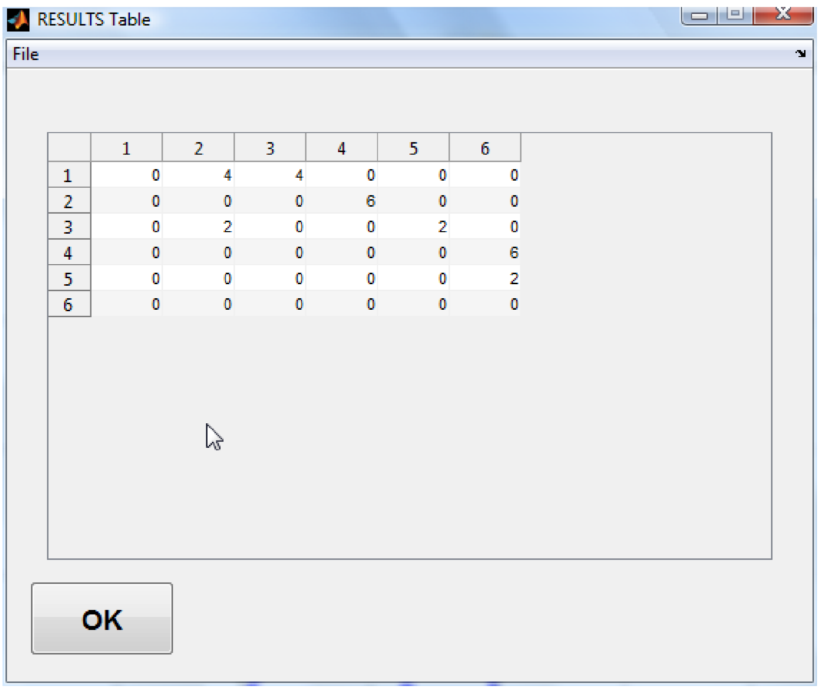
You should now choose Tables to fill in and a Problem to solve!

Figure 4.17: Maximum Flow. Main window after selection of points



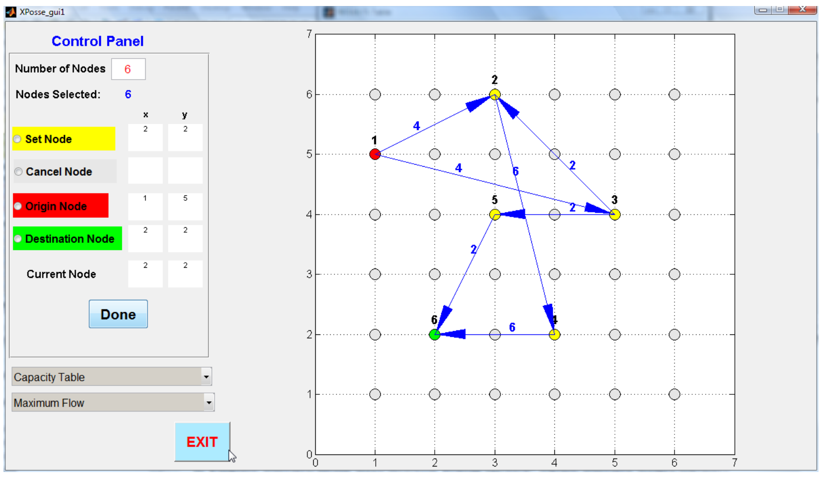
The value of the Objective Function is: 8

Figure 4.18: Maximum Flow. *Results* window



	1	2	3	4	5	6
1	0	4	4	0	0	0
2	0	0	0	6	0	0
3	0	2	0	0	2	0
4	0	0	0	0	0	6
5	0	0	0	0	0	2
6	0	0	0	0	0	0

Figure 4.19: Maximum Flow. *RESULTS Table* window



Number of Nodes 6
Nodes Selected: 6

	x	y
Set Node	2	2
Cancel Node		
Origin Node	1	5
Destination Node	2	2
Current Node	2	2

Capacity Table
Maximum Flow

Figure 4.20: Maximum Flow. Optimum path

Bibliography

[1] `http://www.xposse.eu`

[2] J. Löfberg, YALMIP: A Toolbox for Modeling and Optimization in MATLAB, Proceedings of the CACSD Conference, Taipei, Taiwan, 2004, `http://users.isy.liu.se/johanl/yalmip`.

[3] F. Hillier, G. Lieberman, Introduction to Operations Research, 8th edition, McGraw-Hill Science, 2005.

[4] W. Winston, Operations Research: Applications and Algorithms 4th edition, Duxbury Press, 2003.